FIG.1A

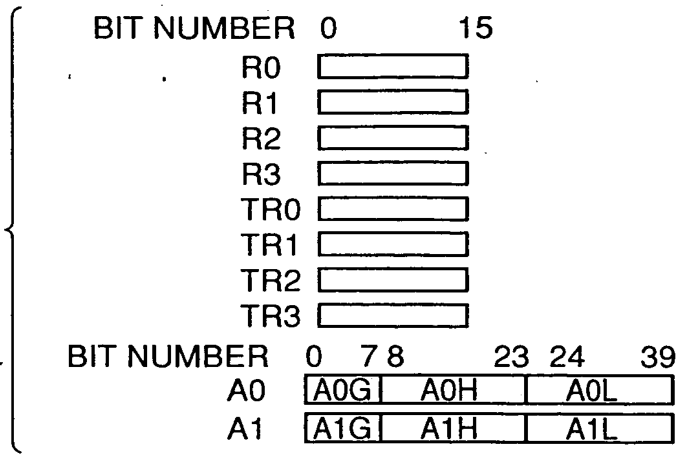

BIT NUMBER 0 15
R0
R1
R2
R3
TR0
TR1
TR2
TR3
BIT NUMBER 0 7 8 23 24 39
A0 A0G A0H A0L
A1 A1G A1H A1L

FIG.1B

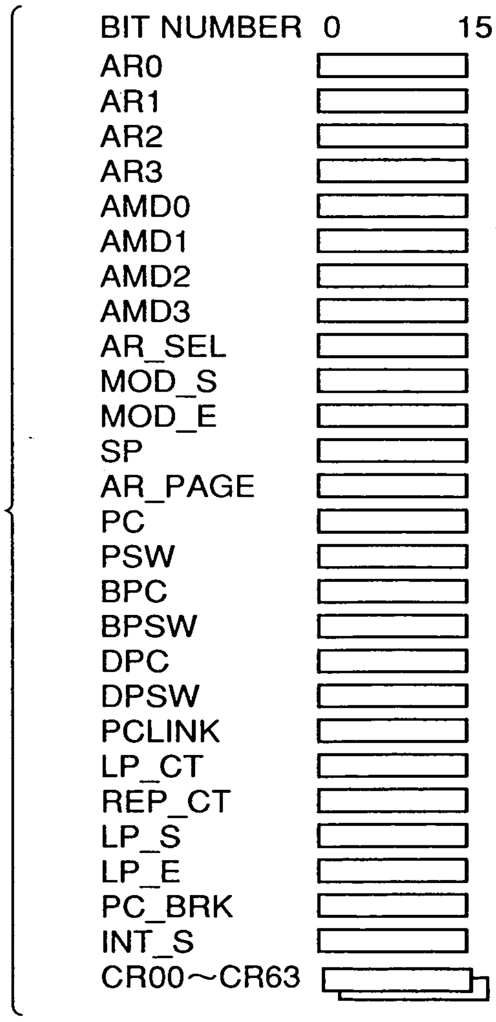
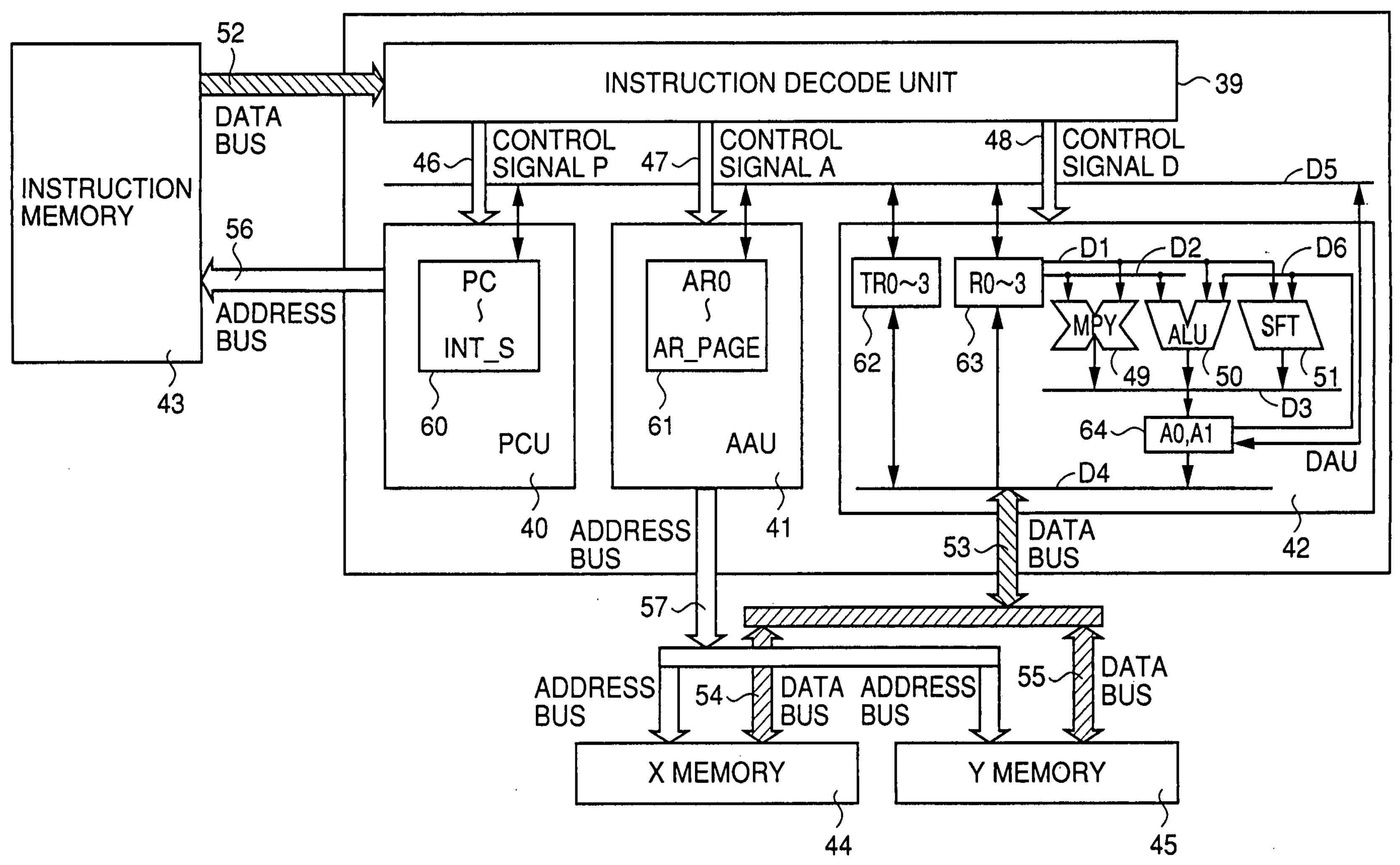

BIT NUMBER 0 15
AR0
AR1
AR2
AR3
AMD0
AMD1
AMD2
AMD3
AR_SEL
MOD_S
MOD_E
SP
AR_PAGE
PC
PSW
BPC
BPSW
DPC
DPSW
PCLINK
LP_CT
REP_CT
LP_S
LP_E
PC_BRK
INT_S
CR00～CR63

FIG.2



INSTRUCTION MEMORY
43
52
DATA BUS
56
ADDRESS BUS
INSTRUCTION DECODE UNIT
39
46
CONTROL SIGNAL P
47
CONTROL SIGNAL A
48
CONTROL SIGNAL D
D5
PC ~ INT_S
60
PCU
40
AR0 ~ AR_PAGE
61
AAU
41
ADDRESS BUS
TR0~3
62
R0~3
63
D1
D2
D6
MPY
49
ALU
50
SFT
51
D3
64
A0,A1
DAU
D4
42
53
DATA BUS
57
ADDRESS BUS
54
DATA BUS
ADDRESS BUS
55
DATA BUS
X MEMORY
44
Y MEMORY
45

FIG.3

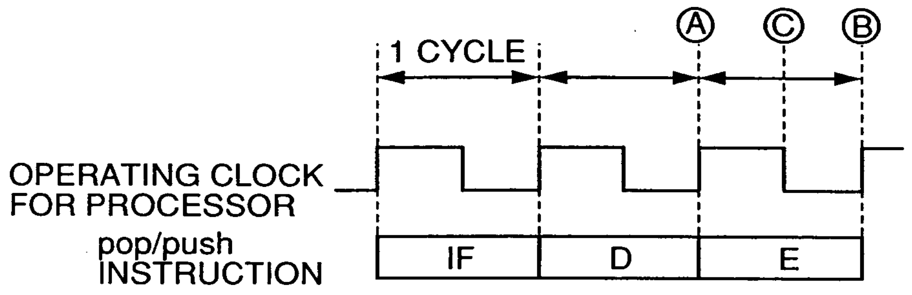

Ⓐ
Ⓒ
Ⓑ
1 CYCLE
OPERATING CLOCK FOR PROCESSOR
pop/push INSTRUCTION
IF
D
E

FIG.4

OPERATION INSTRUCTION	
mul	Multiply
muluu	Multiply unsigned operands
mac	Multiply and add
macuu	Multiply unsigned operands and add
macsu	Multiply signed operand by unsigned operand and add
macsul	Multiply signed operand by unsigned operand and add with shift right
macsuh	Multiply signed operand by unsigned operand and add with shift left
msub	Multiply and sub
msubuu	Multiply unsigned operands and sub
msubsul	Multiply signed operand by unsigned operand and add with shift right
msubsuh	Multiply signed operand by unsigned operand and add with shift left
add	Add a register to acc high
addl	Add a register to acc low
sub	Subtract a register from acc high
subl	Subtract a register from acc low
min	Set minimum value of acch or reg to accumulator
max	Set maximum value of accl or reg to accumulator
amin	Set minimum value to dest-acc
amax	Set maximum value to dest-acc
sra	Shift arithmetic right or left an accumulator
srl	Shift logical right or left an accumulator
and	And
or	Or
xor	Xor
nop	No operation
trfh	Transfer to an accumulator high
trfl	Transfer to an accumulato low
trf	Transfer to an accumulator
aadd	Add accumulators
asub	Subtract src-acc from dest-acc
sadd	Add dest-acc and src-acc with shift
abs	Absolute an accumulator
neg	Negate an accumulator
test	Test an accumulator(acc<0:set Nflag, acc==0:set Zflag)
md	Round an accumulator
not	Not an accumulator

FIG.5

TRANSFER INSTRUCTION

mv	Copy one word from a register to a register
ldi	Load immediate
ld	Load
st	Store
push	Push to stack
put	Put to stack
pop	Pop from stack

SEQUENCE CONTROL INSTRUCTION

jmp	Jump
call	Jump & link
loopi	Set loop counter and start hardware DO loop
loop	Start hardware DO loop
repeati	Set repeat counter and repeat next instruction
repeat	Repeat next instruction
return	Return from subroutine
reit	Return from EIT
rtd	Return from debugger EIT

SPECIAL INSTRUCTION

adr_set	Set AR_SEL register
mvin	Move from IO registers
mvout	Move to IO registers
slave	Transit to slav mode
noop	No operation

FIG.6

	ADDRESSABLE REGISTER
LOAD INSTRUCTION	R0, R1, R2, R3, TR0, TR1, TR2, TR3
STORE INSTRUCTION	TR0, TR1, TR2, TR3, A0H, A0L, A1H, A1L

FIG.7

```
LDI  AR3, #STACK_BOTTOM      ; (1)
LDI  AMD3, #DEC_1            ; (2)
  • • • • • • •
ST  TR0, X:AR3               ; (3)
MV  TR0, AR0                 ; (4)
ST  TR0, X:AR3               ; (5)
```

FIG.8A FIG.8B FIG.8C FIG.8D

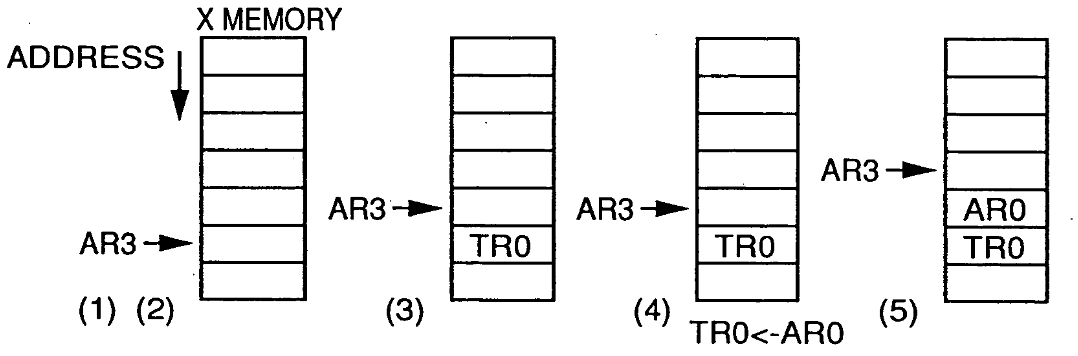


FIG.9A

```
POP
[mnemonics]
            (1) pop
            (2) pop  ra
[operation]
            (1) tr0 = x_memory[sp];
                  sp++;
            (2) ra = tr0;
                  tr0 = x_memory[sp];
                  sp++;
```

FIG.9B

```
PUSH
[mnemonics]
            (1) push
            (2) push  ra
[operation]
            (1)
                  sp--;
            (2) x_memory[sp] = tr0;
                  tr0 = ra;
                  sp--;
```

FIG.9C

```
PUT
[mnemonics]
             put
[operation]
            x_memory[sp] = tr0;
```

FIG.10

```
push               ; (1)
push  R0           ; (2)
push  AR0          ; (3)
put                ; (4)
. . . . . . . . . . .
pop                ; (5)
pop  AR0           ; (6)
pop  R0            ; (7)
```

FIG.11A

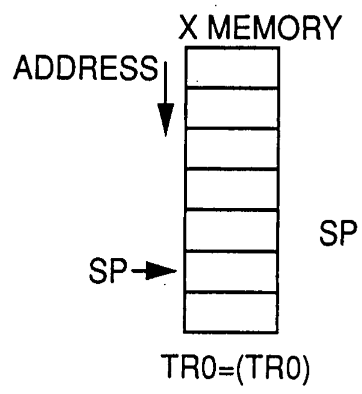


FIG.11B

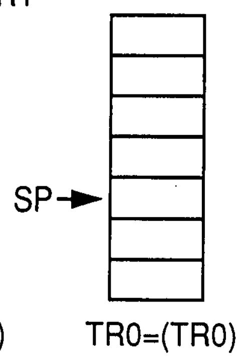


FIG.11C

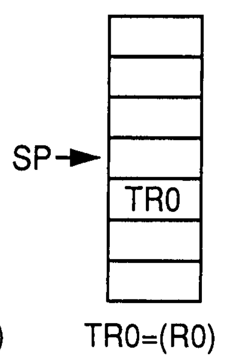


FIG.11D

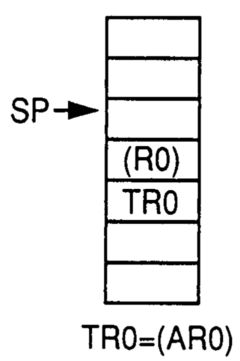


FIG.11E

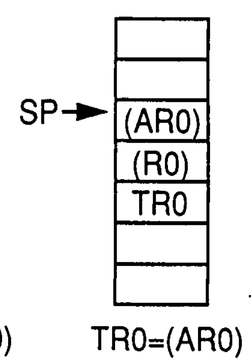


FIG.11F

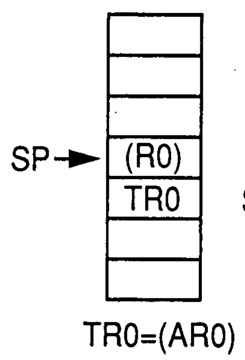


FIG.11G

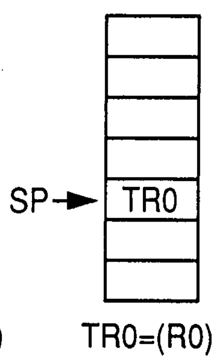


FIG.11H

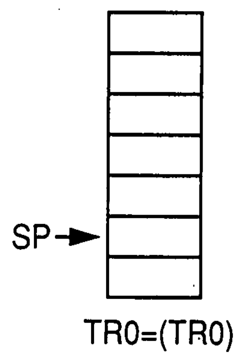


FIG.12

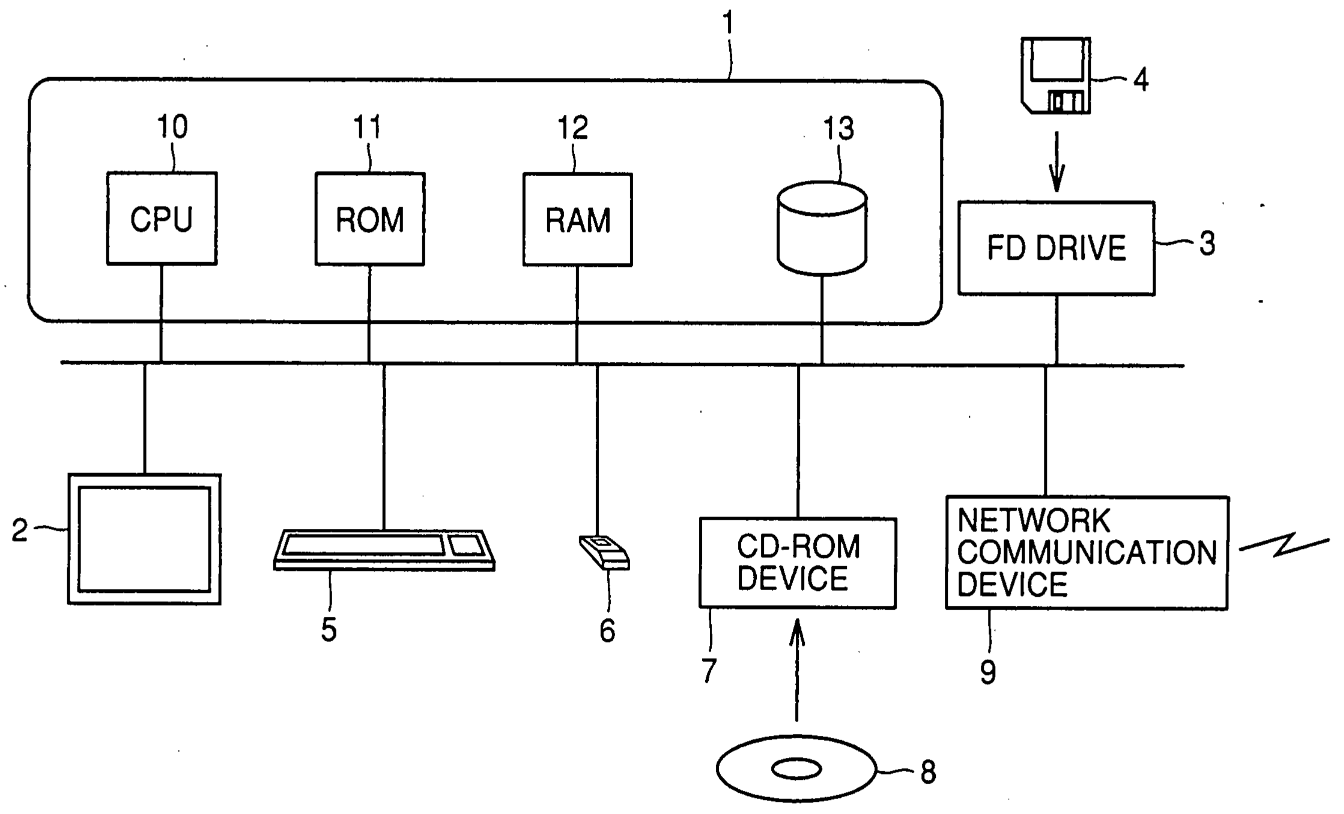

1
10
CPU
11
ROM
12
RAM
13
FD DRIVE
3
4
2
5
6
CD-ROM DEVICE
7
8
NETWORK COMMUNICATION DEVICE
9

FIG.13A

```
MPUSH  R0, AR0; (1)
. . . . . . . . . .
MPOP   ;(2)
```

FIG.13B

```
push        ;
push  R0    ;
push  AR0   ;
put         ;
```

FIG.13C

```
pop         ;
pop  AR0    ;
pop  R0     ;
```

FIG.14

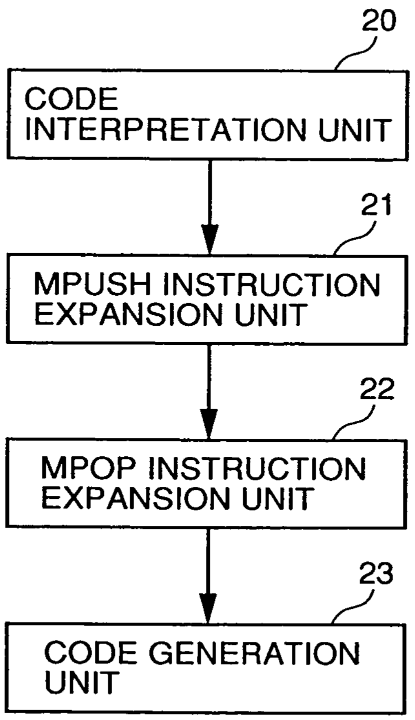

20
CODE INTERPRETATION UNIT
21
MPUSH INSTRUCTION EXPANSION UNIT
22
MPOP INSTRUCTION EXPANSION UNIT
23
CODE GENERATION UNIT

FIG. 15

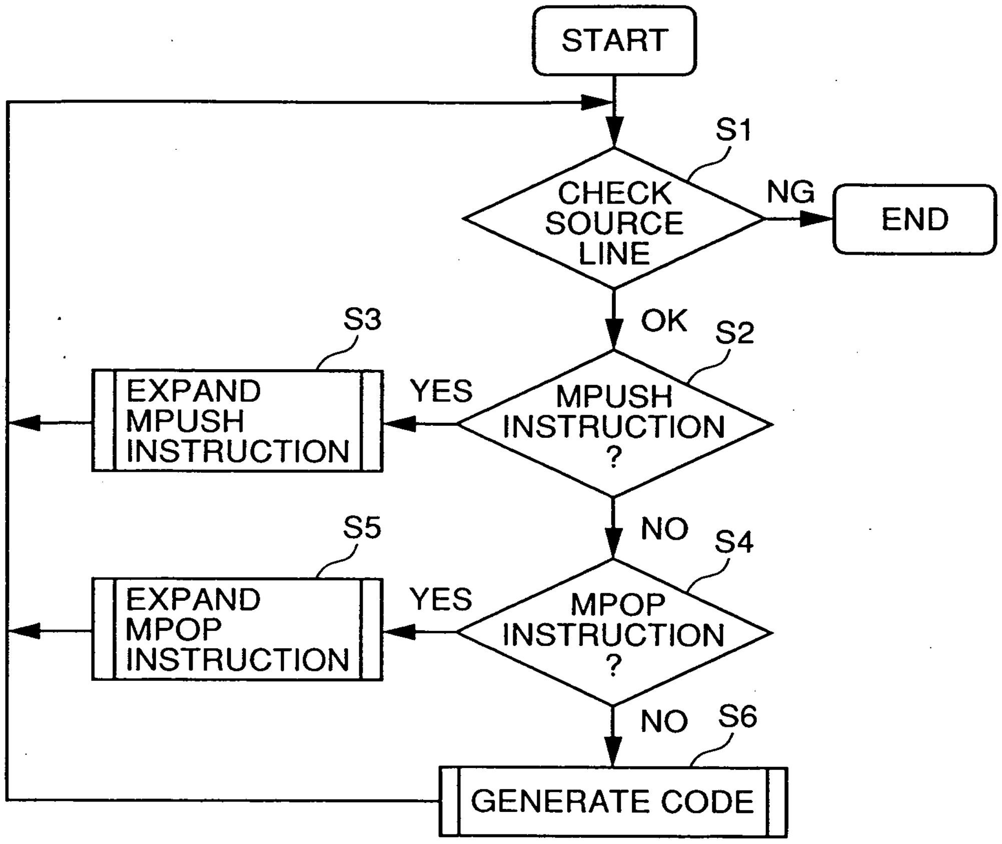

START
S1
CHECK SOURCE LINE
NG
END
OK
S2
MPUSH INSTRUCTION ?
YES
S3
EXPAND MPUSH INSTRUCTION
NO
S4
MPOP INSTRUCTION ?
YES
S5
EXPAND MPOP INSTRUCTION
NO
S6
GENERATE CODE

FIG. 16

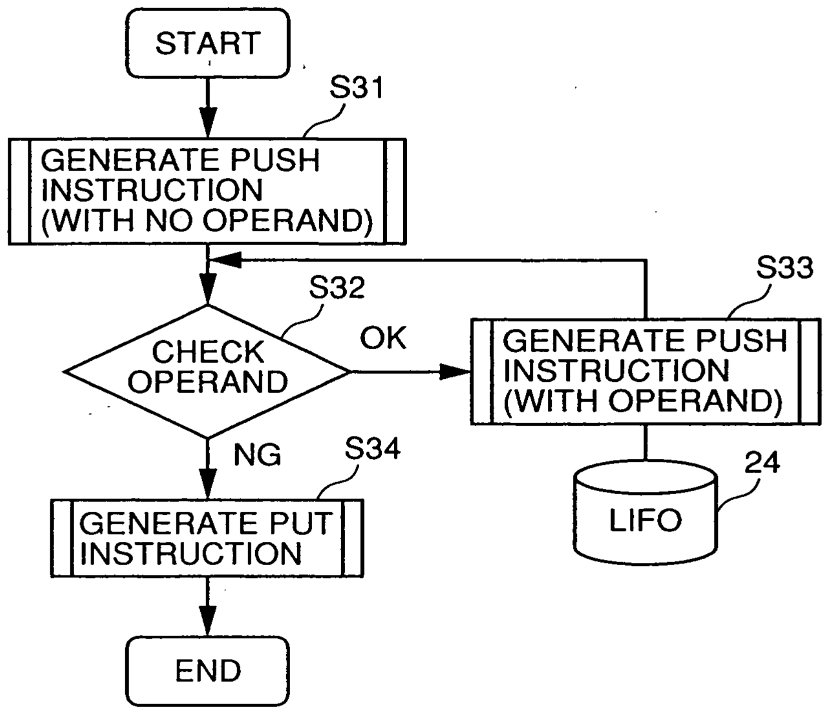

START
S31
GENERATE PUSH INSTRUCTION (WITH NO OPERAND)
S32
CHECK OPERAND
OK
S33
GENERATE PUSH INSTRUCTION (WITH OPERAND)
NG
S34
GENERATE PUT INSTRUCTION
24
LIFO
END

FIG. 17

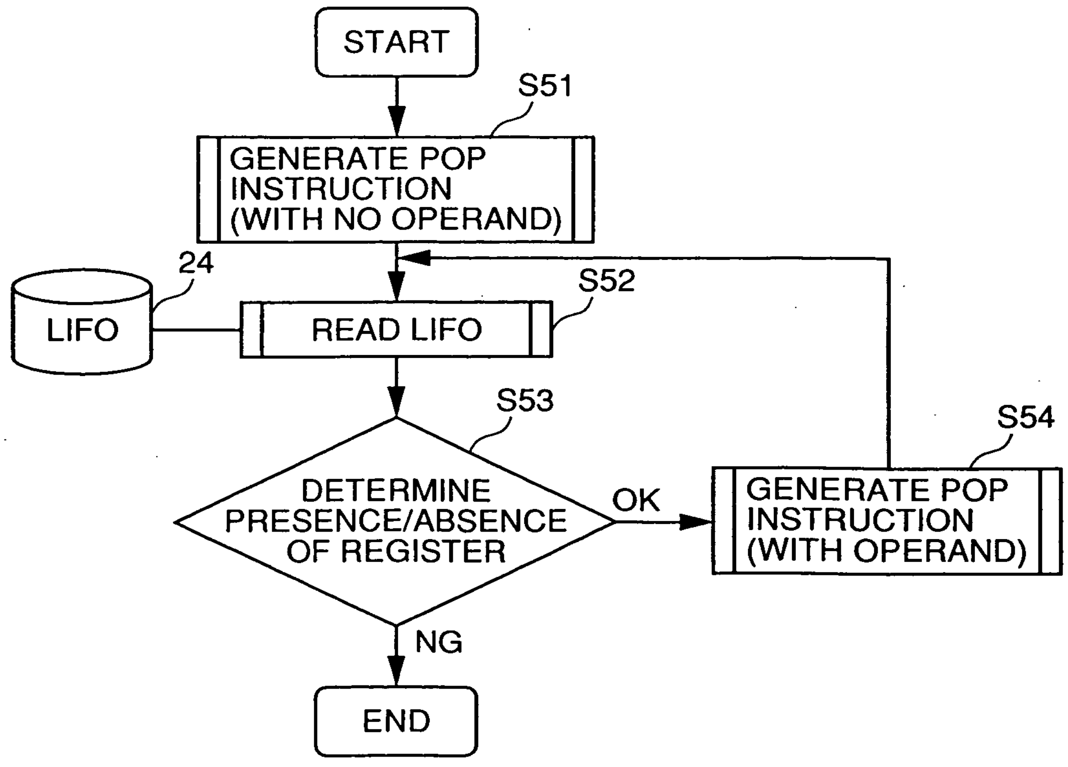

START
S51
GENERATE POP INSTRUCTION (WITH NO OPERAND)
24
LIFO
S52
READ LIFO
S53
DETERMINE PRESENCE/ABSENCE OF REGISTER
OK
S54
GENERATE POP INSTRUCTION (WITH OPERAND)
NG
END

FIG.18A

```
MPUSH  TR0, AR0; (1)
. . . . . . . . . .
MPOP   ;(2)
```

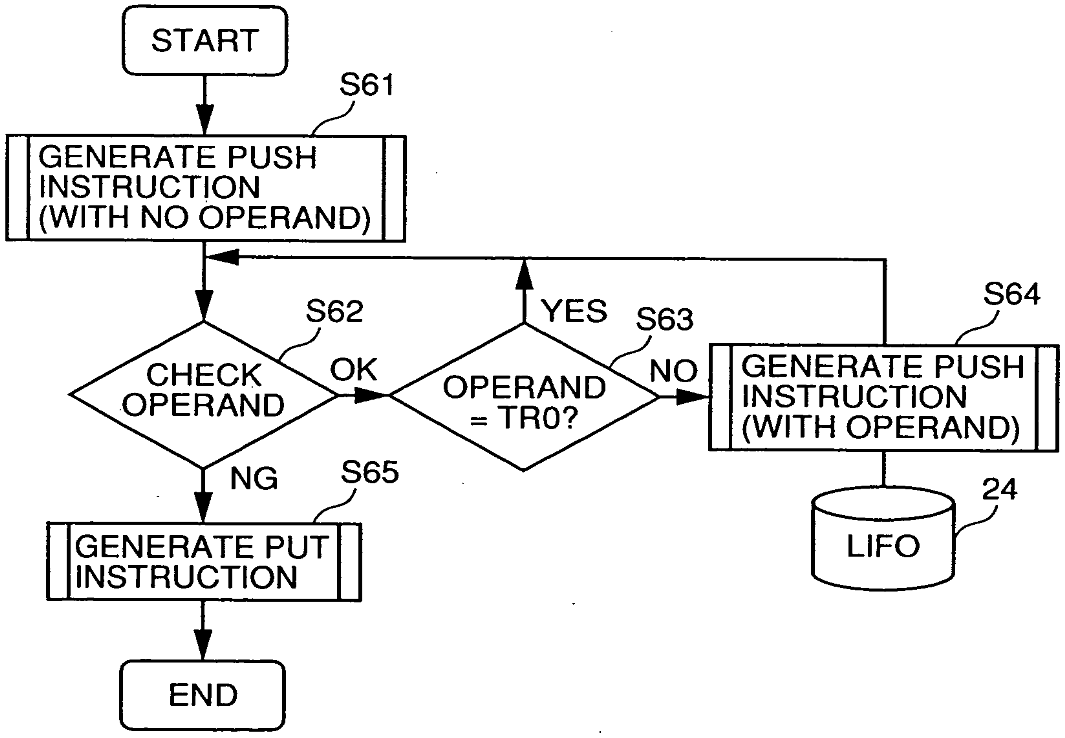
FIG.18B

```
push        ;
push  AR0   ;
put         ;
```

FIG.18C

```
pop         ;
pop  AR0    ;
```

FIG.19



START
S61
GENERATE PUSH INSTRUCTION (WITH NO OPERAND)
S62
CHECK OPERAND
OK
NG
YES
S63
OPERAND = TR0?
NO
S64
GENERATE PUSH INSTRUCTION (WITH OPERAND)
24
LIFO
S65
GENERATE PUT INSTRUCTION
END